車でモダンなデジタル体験を可能にする "Cloud-Native Vehicle" コンセプト

基調講演 @ OSAKA NDS Embedded Linux Cross Forum No.10

宗像尚郎

Automotive Grade Linux Advisory board member
Linux Foundation board member

2020-2-5

自己紹介

ルネサスエレクトロニクス という半導体メーカーで働いています

- ルネサスの概要と私の担当業務
  - ルネサスエレクトロニクス=（日立+三菱）+ NEC の半導体事業部門
  - 車載向け SOC（System on Chip) のグローバルリーディングサプライヤーです
  - R-CarH3 = 64bit ARM Octa-core + GPU + LPDDR4 8G、1,384pin BGA
  - 私は SOC を動かすために必要な Linux BSP の開発や提供を担当 しています
- 産業コンソーシアム活動、コミュニティ活動でも指導的な役割をもっています
  - Linux Foundation（後述）の全世界で 22 名の理事メンバー の一人です
  - Automotive Grade Linux、yocto project の理事メンバーです
  - 社内オープンソース開発チームを運営、コミュニティ開発でも顕著な実績

ルネサスはオープンソース開発活動への貢献実績において日本を代表する企業です

Linux kernel 開発実績（kernel 5.3, 2019-9-16 release の実績）

Most active 5.3 employers

By changesets

Intel	1545	10.7%
AMD	966	6.7%
(Unknown)	963	6.7%
Red Hat	925	6.4%
(None)	865	6.0%
Google	638	4.4%
Renesas Electronics	638	4.4%
Linaro	548	3.8%
IBM	460	3.2%
Mellanox	440	3.0%
(Consultant)	410	2.8%
Huawei Technologies	372	2.6%
SUSE	337	2.3%
Samsung	311	2.2%
NXP Semiconductors	309	2.1%
Linux Foundation	291	2.0%
ARM	280	1.9%
Facebook	188	1.3%
Oracle	170	1.2%
BayLibre	170	1.2%

By lines changed

AMD	432537	37.0%
Linaro	115926	9.9%
Intel	76950	6.6%
Mellanox	46732	4.0%
Samsung	37869	3.2%
Google	31666	2.7%
(Unknown)	31310	2.7%
Red Hat	31089	2.7%
IBM	29449	2.5%
(None)	27803	2.4%
SUSE	18928	1.6%
Cirrus Logic	16509	1.4%
(Consultant)	14192	1.2%
Facebook	13723	1.2%
Linux Foundation	13405	1.1%
ARM	12755	1.1%
Pengutronix	12398	1.1%
NXP Semiconductors	12202	1.0%
Renesas Electronics	11345	1.0%
Huawei Technologies	10165	0.9%

Most active 5.3 developers

By changesets

Kuninori Morimoto	271	1.9%
Christoph Hellwig	262	1.8%
Mauro Carvalho Chehab	205	1.4%
Nishka Dasgupta	162	1.1%
Chris Wilson	160	1.1%
Greg Kroah-Hartman	145	1.0%
Yue Haibing	137	0.9%
Masahiro Yamada	130	0.9%
Gustavo A. R. Silva	128	0.9%
Geert Uytterhoeven	120	0.8%
Takashi Sakamoto	113	0.8%
Wolfram Sang	101	0.7%
Arnaldo Carvalho de Melo	99	0.7%
Colin Ian King	98	0.7%
Arnd Bergmann	96	0.7%
Adrian Hunter	96	0.7%
Russell King	94	0.7%
David Howells	92	0.6%
Hawking Zhang	90	0.6%
Maxime Ripard	84	0.6%

By changed lines

Hawking Zhang	364371	31.2%
Arnd Bergmann	84993	7.3%
Mauro Carvalho Chehab	34041	2.9%
Harry Wentland	33568	2.9%
Jason Gunthorpe	22421	1.9%
Chris Wilson	17382	1.5%
Richard Fitzgerald	15791	1.4%
Linus Walleij	13392	1.1%
Greg Kroah-Hartman	11713	1.0%
Bernard Metzler	10963	0.9%
Christoph Hellwig	10844	0.9%
Pawel Laszczak	7702	0.7%
Hannes Reinecke	7255	0.6%
Andrii Nakryiko	7143	0.6%
Maxime Ripard	6709	0.6%
David Ahern	6557	0.6%
Daniele Ceraolo Spurio	5781	0.5%
Darrick J. Wong	5776	0.5%
Kuninori Morimoto	4717	0.4%
Thomas Zimmermann	4706	0.4%

https://lwn.net/Articles/798505/

Linux kernel 開発実績（2005 年 4 月からの 14 年間強の累積）

Total patch sets of this kernel release: 822303
888 companies contribute their works to this kernel release.
Averagely, every companies committed 926 patch sets.

No.1	Unknown	115707(14.07%)
No.2	Intel	84981(10.33%)
No.3	Red Hat	72879(8.86%)
No.4	Hobbyists	71581(8.70%)
No.5	Novell	35687(4.34%)
No.6	IBM	32948(4.01%)
No.7	Linaro	24061(2.93%)
No.8	AMD	17729(2.16%)
No.9	Google	16856(2.05%)
No.10	Renesas Electronics	16624(2.02%)
No.11	Samsung	16534(2.01%)
No.12	Oracle	16006(1.95%)
No.13	Texas Instruments	15182(1.85%)
No.14	Mellanox Technologies	9718(1.18%)
No.15	Academics	8311(1.01%)
No.16	HuaWei	8125(0.99%)
No.17	Consultants	8020(0.98%)
No.18	ARM	7815(0.95%)
No.19	Broadcom	7578(0.92%)
No.20	Bootlin	7013(0.85%)
No.21	Linutronix	6499(0.79%)
No.22	Linux Foundation	6092(0.74%)
No.23	VISION Engraving and Routing Systems	6045(0.74%)
No.24	Canonical	5976(0.73%)
No.25	Pengutronix	5679(0.69%)
No.26	Analog Devices	5536(0.67%)
No.27	NXP	5466(0.66%)
No.28	NVIDIA	5420(0.66%)
No.29	Fujitsu	5076(0.62%)
No.30	QUALCOMM	4902(0.60%)
No.31	Code Aurora Forum	4785(0.58%)
No.32	Freescale	4694(0.57%)
No.33	Wolfson Microelectronics	4180(0.51%)
No.34	Nokia	4072(0.50%)
No.35	Cisco	4021(0.49%)
No.36	Parallels	3805(0.46%)
No.37	Imagination Technologies	3773(0.46%)
No.38	Marvell	3635(0.44%)
No.39	QLogic	3390(0.41%)
No.40	NetApp	2827(0.34%)

http://www.remword.com/kps_result/all_whole.html

つながるクルマの世界を実現していくために

クルマだけがコネクテッドに乗り遅れている？

windows95 がインターネットを全ての人に解放した転換点となった

1995 年以降に PC/Cloud が達成してきた成果、直面した課題

- Windows95 登場により だれでもインターネットを利用できる ようになった
- ただちに スパムメールやコンピュータウィルス の問題が社会問題化
- ソフトウエア更新サービス がスタート（Windows Update は win95 から）
- mp3 音源による デジタル携帯音楽プレーヤ（初代 iPod は 2001 年）
- 仮想化技術（Xen は 2003 年、KVM は 2006 年）
- クラウド利用の オンラインストレージ （Dropbox は 2008 年にサービス開始）
- スマートフォン の登場によるデジタルデータの活用促進（iPhone は 2007 年）
- SaaS（Software as a Service） （Office 365 は 2011 年にスタート）

過去四半世紀に確立されたデジタル技術が、今一気にクルマの世界に波及してきている

スマフォが当たり前の人々が Connected Car に求める期待値

ユーザーの期待は HMI の見た目を似せることなんかじゃない！

- ユーザーのコンテキスト（= 事情）を逐次解釈した賢いガイドをして欲しい
  - 運転中に「進行方向、車線」や「目的地」を考慮したガイダンスができること
  - 個人の「オンラインスケジュール」を参照 して目的地候補を提案できる
- AI アシスタント / エージェント機能 (Siri, Google アシスタント)
  - 自然な話し言葉でインタラクティブにクルマとコミュニケーションできる
- Twitter, Facebook, Instagram などソーシャルアプリとのシームレス連携
  - SMS メッセージの送受信, 写真アップロード、..
  - メッセージのブロードキャスト（=同報）, 自車位置の共有,..
- ソフトウエア (= アプリ) のインストール／アップデート によるカスタマイズ

既にスマフォの世界観に慣れ親しんでいるユーザーの期待値に応えていく必要がある

コネクテッドカー普及による 新サービスビジネス への期待も大

車両情報と最新の Cloud/AI 技術の融合による最新デジタル経験の提供 が狙い

- ソフトウエア領域
  - 保険業＝ 運転状況連動型保険（=UBI)
  - レンタカー業者＝ 車両の運行管理システム （遠隔操作、オンライン契約管理）
  - 宅急便＝自家用車のトランクを臨時の宅配 BOXとして活用するサービス
  - レンタルビデオ＝契約している音楽、映像配信サービスを車内でも利用
- ハードウエア領域
  - 高性能ドライブレコーダー ＝クラウド連携、イベント通知機能
  - 故障診断装置＝ AI を活用した知識ベースに基づく故障解析、予知診断
  - フリートの運行管理＝ デジタルタコグラフ（=EDL)
  - デジタルビーコン ＝クルマと社会インフラを結び付けるための情報送信ボックス

どこまで車両情報が開示されるか次第ではスマートフォン以上の価値提供の可能性も

何故クルマだけが コネクテッドから取り残されてきた のだろうか

通信モデムが搭載されていなかった.... のは結果であって原因ではない?

- 既存の通信プロトコルやシステムは高速に移動するクルマでは利用しにくい
  - TCP/IP プロトコル では IP アドレス自体に位置情報が含まれている
  - 一時的なネットの遮断 があっても車載アプリケーションは止まってはいけない
  - 車両やドライバーを特定するための 標準的な ID 識別手段 がなかった
  - よほど重大な SW 不具合対策以外では 車両販売後に SW を更新する必要はなかった
- スマフォや PC とは全く異なる信頼性のレベルが求められる
  - (一部の) クルマの機構が故障してしまうと 生命の危機に直結 するリスク
  - 大規模なメカトロニクス全体で信頼性を担保しているので部分的な機能更新は困難
  - アプリケーションの素性によりどこまで車両情報を見せ制御を許すかに差をつける

クルマは元々は自己完結したクローズドシステム、コネクテッドはオープン化への挑戦

コネクテッドカー実現の課題

課題１：ソフトウエアの進化（更新）のスピードに追従できない

（車載機に限らず）組み込み機器では 製品購入後には SW を変更できないのが普通

- PC / IT / スマートフォン
  - ハードウエアベンダーは基本ソフト（=OS）をインストールした状態で出荷
  - ユーザーは用途に応じてアプリケーションをインストールしてカスタマイズ可能
  - ネット経由で SW のバージョンアップを受け取ることができ、更新作業も容易
- クルマ（や、他の組み込み機器全般）
  - アプリケーションまで含め 全ての SW を組み込んだ状態で製品を出荷
  - 車載機の基本ソフトの更新は容易ではなく 都度 Tier1 に更新してもらう必要 がある
  - ユーザーは アプリ単位で SW を更新したりロールバックすることは出来ない
  - 近年 サイバーセキュリティ対策 のための部分更新や全体更新が要請されている

スマフォの頻繁なアプリ更新（=サービスメニューの追加）には対応できていない

課題２：車載向けアプリケーション開発者が確保できない

Web アプリ開発者人口　組み込み SW 開発者人口 が制約要因になっている

- Web (html5) アプリケーション開発
  - Javascript、Python、Java など生産性の高いプログラミング言語を利用
  - 優れた統合開発環境、サンプルコードなど PC 上でアプリ開発が完結
  - アジャイル開発（逐次機能追加）、DevOps（開発と運用の連携） 等の開発スタイル
  - 大規模クライアントの同時更新に対応可能 な優れた SW 配信メカニズムを利用可能
- 車載機アプリケーション開発
  - C/C++などターゲットハードウエアアーキテクチャ寄りのプログラミング言語
  - 専用の評価ボード、実車との接続 などアプリケーション評価の敷居が高い
  - 単体機能モジュールの インテグレーション後の結合評価に多くの時間 がかかる

最先端 IT/Cloud/AI 技術者から見ると車載機向け SW 開発はとてもハードルが高い

課題３：クルマ特有の情報（CAN など） が開示されていない

クルマの信号インターフェースは 当然各社ユニークで PC との互換性は無い

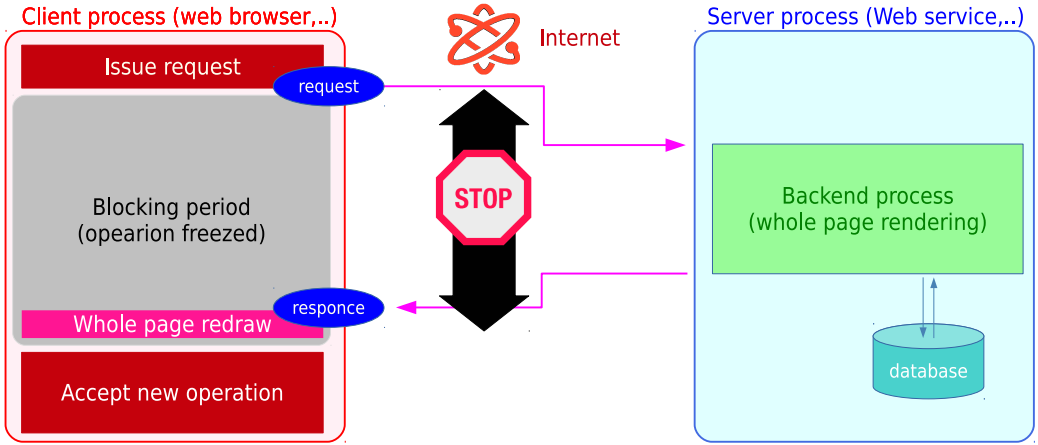
- PC / IT 領域
  - 汎用計算機、ミニコン、ワークステーション時代は各社独自インターフェース
  - PC/AT 互換機 の登場以降、プリンター等の接続インターフェースが共通化された
  - 近年は USB、Ethernet、HDMI/DVI など標準的な物理インターフェースが普及
- クルマ
  - 振動や温度、長時間動作に耐える車載専用の高信頼信号コネクターを採用
  - 電源コネクターなどの 物理形状や信号配置は各自動車メーカーの独自仕様
  - 車載式故障診断装置（OBD-2 = On-Board Diagnostics) データーリンクコネクター
    外形は標準だが、汎用診断コード（公開）以外に独自コード（非公開）がある
  - 現状 OBD-2 以外に車両情報を取り出すための物理インターフェースはない

PC 向け周辺機器とは異なりクルマ向けの汎用後付け BOX 開発の可能性はごく限定的

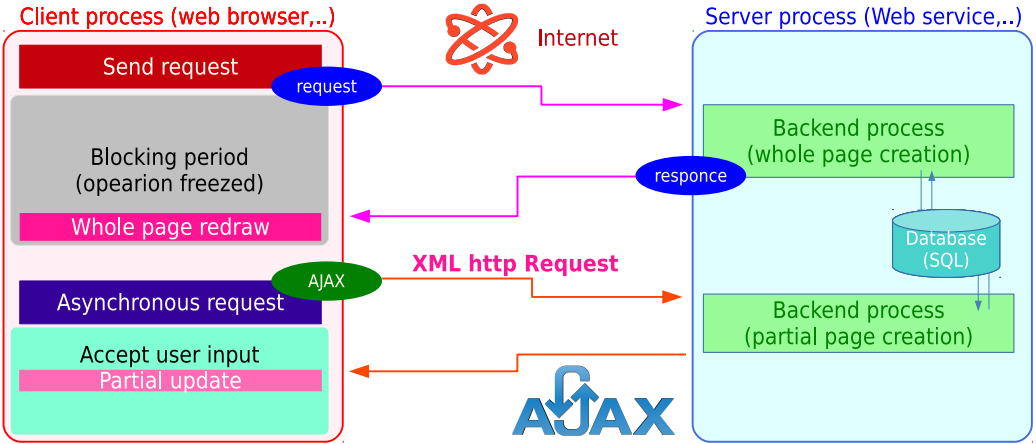
コンピュータの歴史を振り返ってみよう

サーバー間連携の変遷を振り返る (ブロッキング型 -> 非同期型)

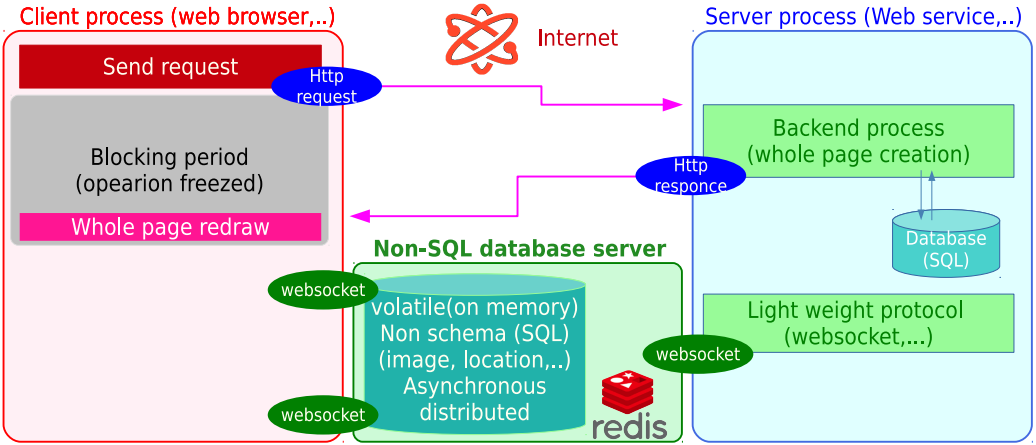
2010 年頃までのサーバー呼び出しは基本的に「ブロッキング」



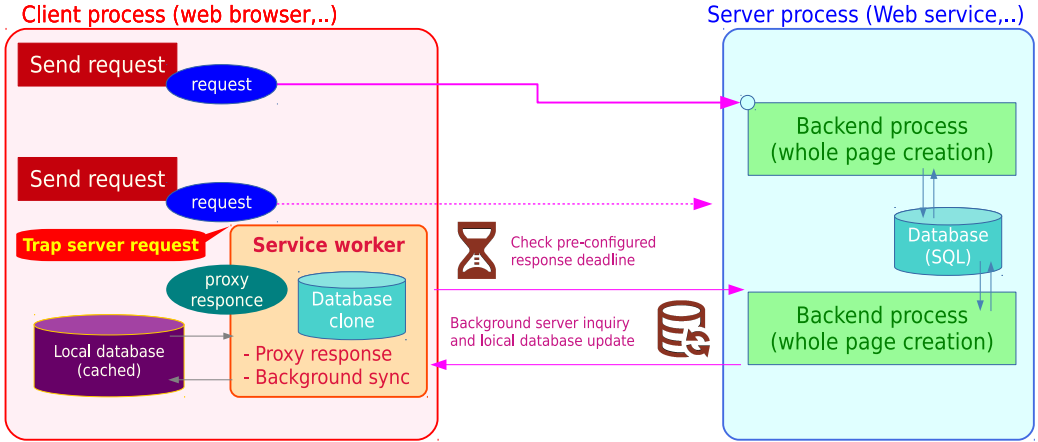
非同期インターフェース (1) Asynchronous Javascript And XML



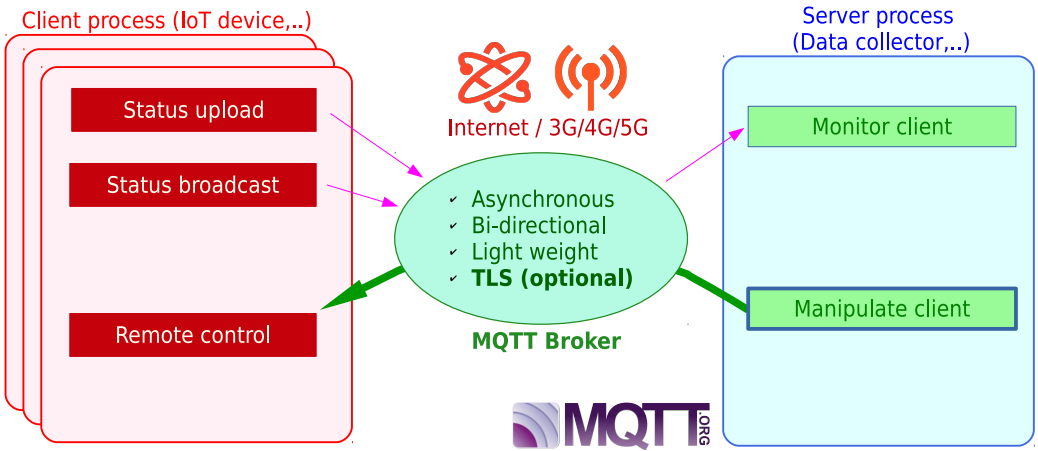
非同期インターフェース (2) Non-SQL, on-memory データーベース



非同期インターフェース (3) service worker (proxy 応答)

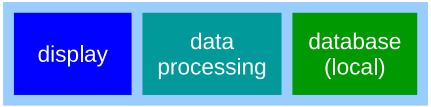


非同期インターフェース (4) MQTT (Message Q. Telemetry Transport)

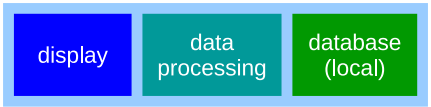


近年「つながる」カーナビが増えてきたが...

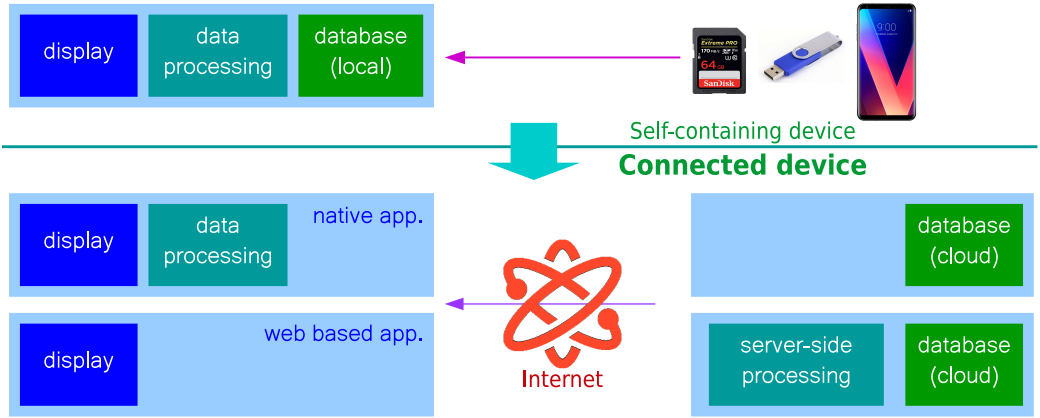
カーナビのトレンド (1) 自己完結



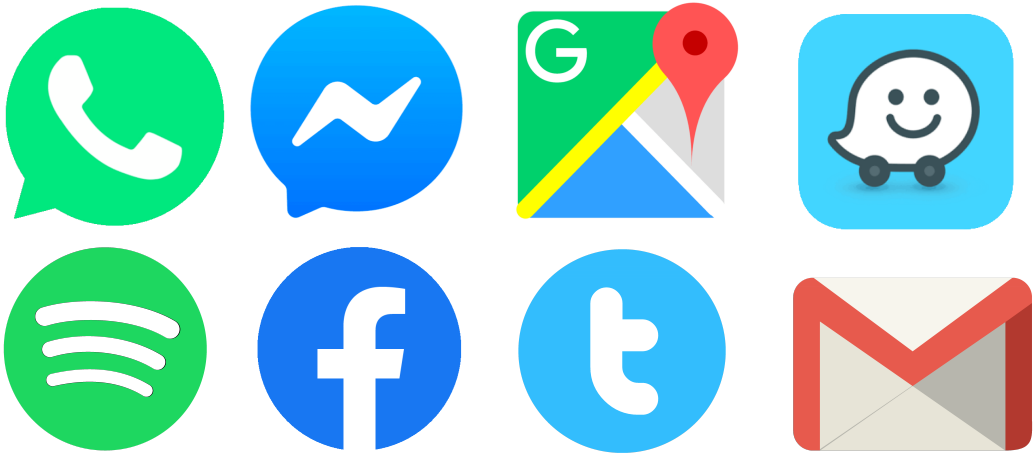
カーナビのトレンド (2) 自己完結 + 外部コンテンツの利用



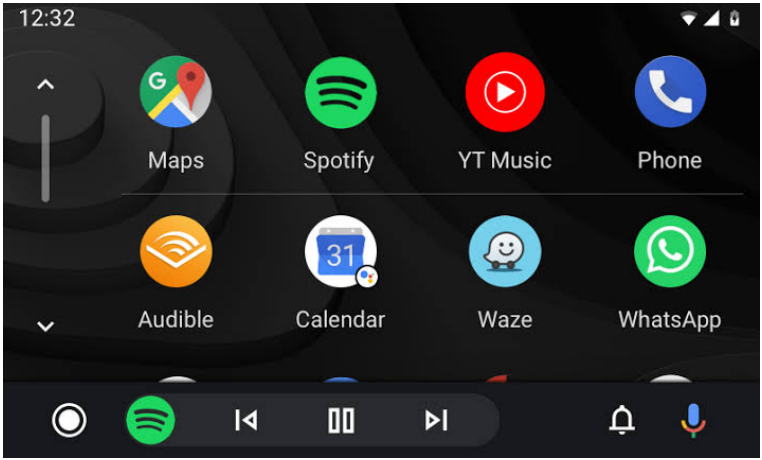
カーナビのトレンド (3) コネクテッドデバイス



モバイルアプリの大部分は 非同期なクラウド連携 を利用したもの



"Android Auto" には "Cloud Native 機能" が既に統合されている

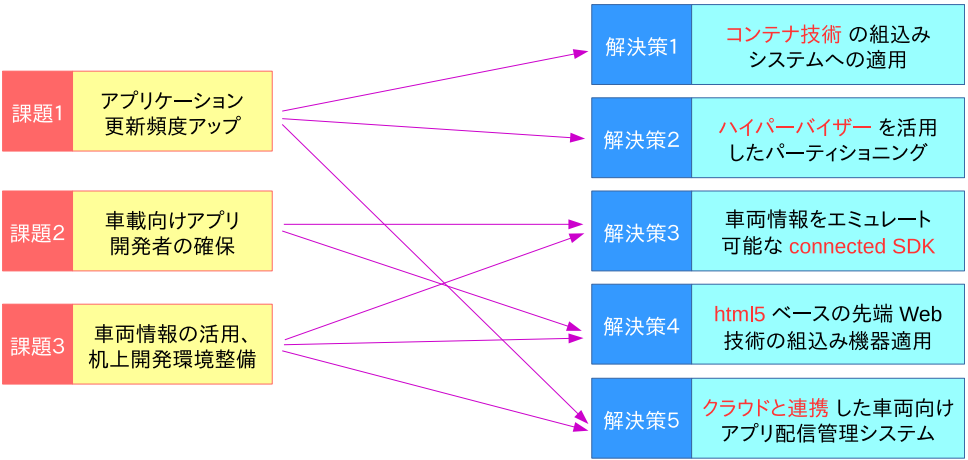


Android が対応している機能

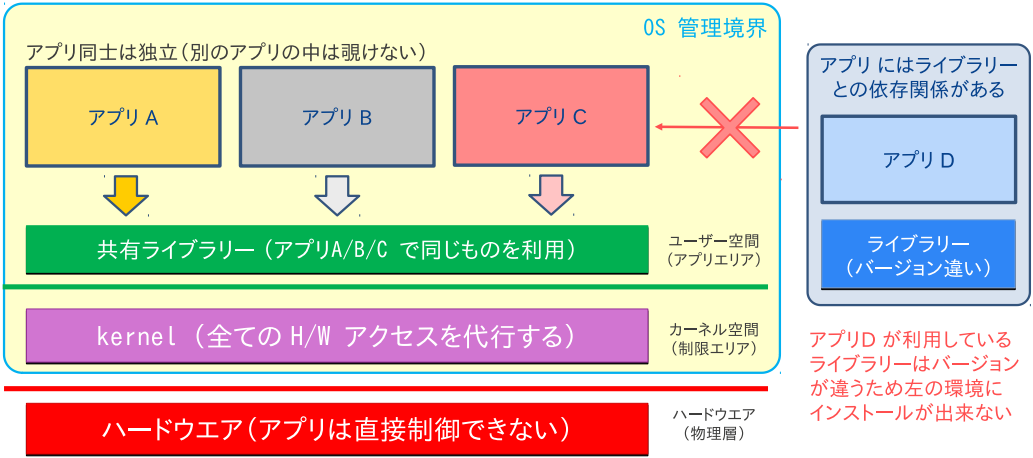
- シームレスなアプリ間のデータ連携 (native + cloud)
- 非同期クラウド連携
- App ストア
- 機器間データ連携 (PC, 電話, クルマ)

クルマでも利用可能となってきた最新 SW 技術

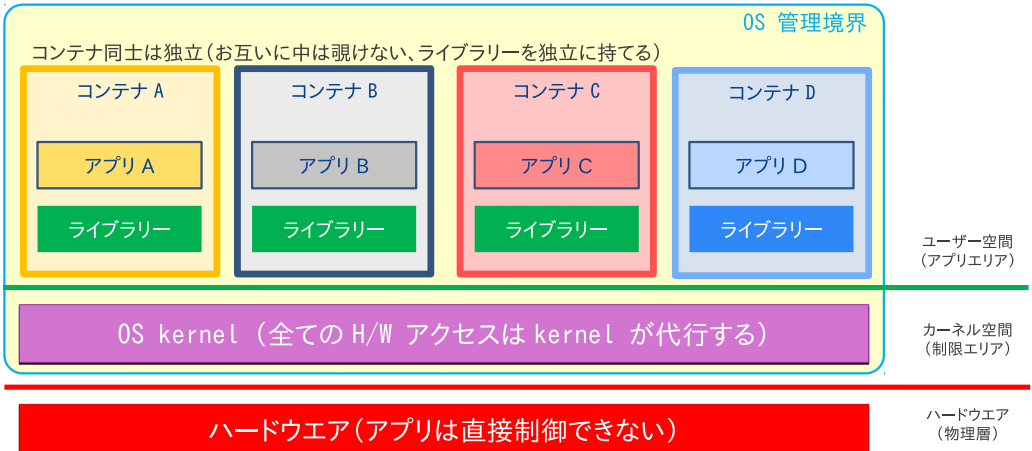
コネクテッドカー実現の諸課題に対する 解決策



解決策 0 : 標準的な Linux の構成 (= 現在の車載機の SW 構成)



解決策１：コンテナ技術 （ソフトウエア抽象化）



解決策２：ハイパーバイザー技術 （ハードウエア抽象化）

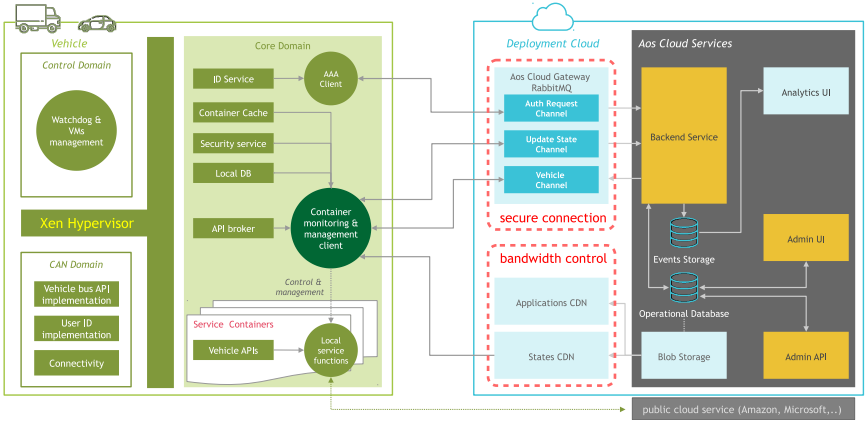
解決策３：コネクテッドカー用ソフトウェア開発ツール（SDK）

← 車両シミュレータ

- ツールからクルマの操作（運転操作、ドア／トランク開閉など）に対応した車両情報に該当するW3C VISSコマンドが発行できる
- 手軽な PC 上でコネクテッドカーアプリケーションの提案や試作が可能に
- 車載アプリ 開発者の敷居の大幅ダウン に寄与

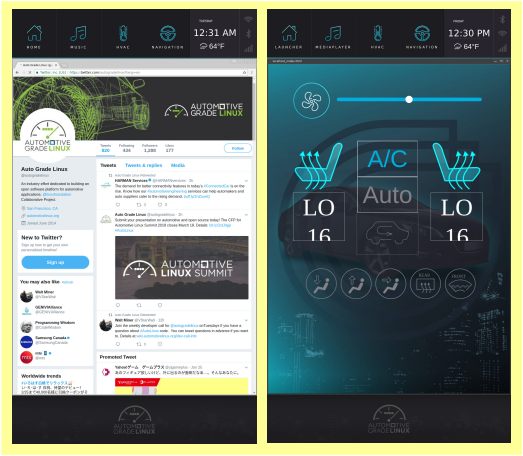
https://www.renesas.com/jp/ja/about/press-center/news/2018/news20181016.html

解決策４：クラウド連携 (Renesas/EPAM AoS Function Management)



https://bit.ly/2DyN9OD

解決策５：Renesas R-CarH3 上で Chromium ブラウザー が動作



Chromium on Renesas R-CarH3

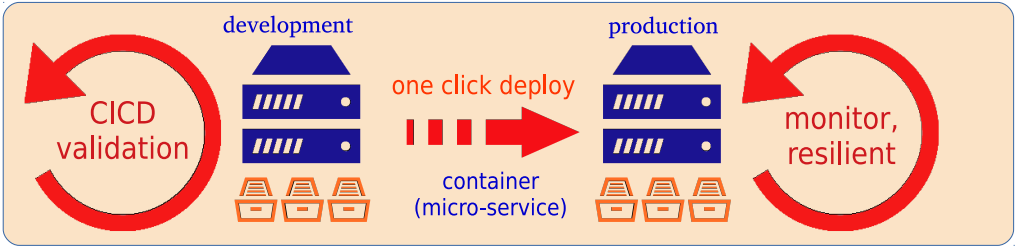
- Renesas 製、車載機制御用高性能 SOC
- ARM 64bit CPU x8（マルチコア）
- 高性能グラフィックスエンジン搭載
- マルチメディアデコーダー内蔵

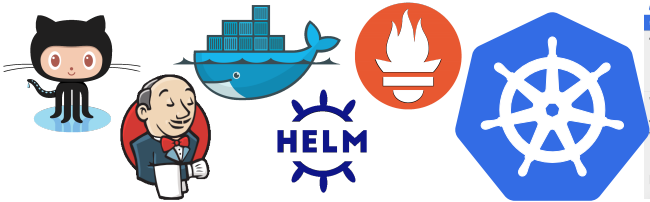
- Upstream Linux kernel 標準サポート
- Chromium Upstream 追従中
- Wayland Window Manager 対応
- AGL（Automotive Grade Linux）

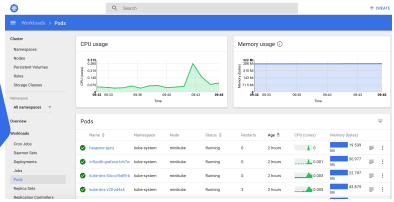
コネクテッドカー!= Cloud Native Vehicle

更なるクラウドとの連携による Cloud Native システムの実現

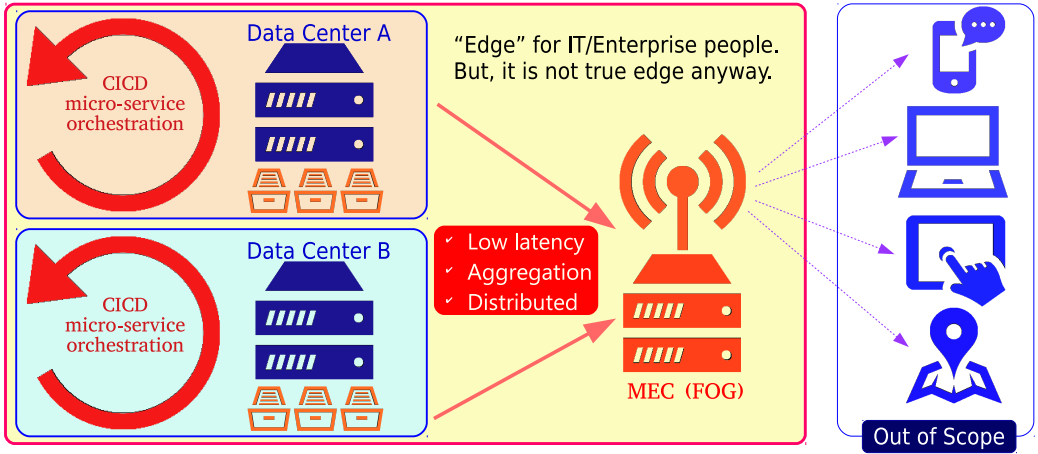
"Cloud Native" は現状は IT/エンタープライズ業界特有 のトピック







テレコム業界は MEC (=Mobile Edge Computing) を提案中



コネクテッド と Cloud Native は何が違うのか

コネクテッド = 同期（ブロッキング）

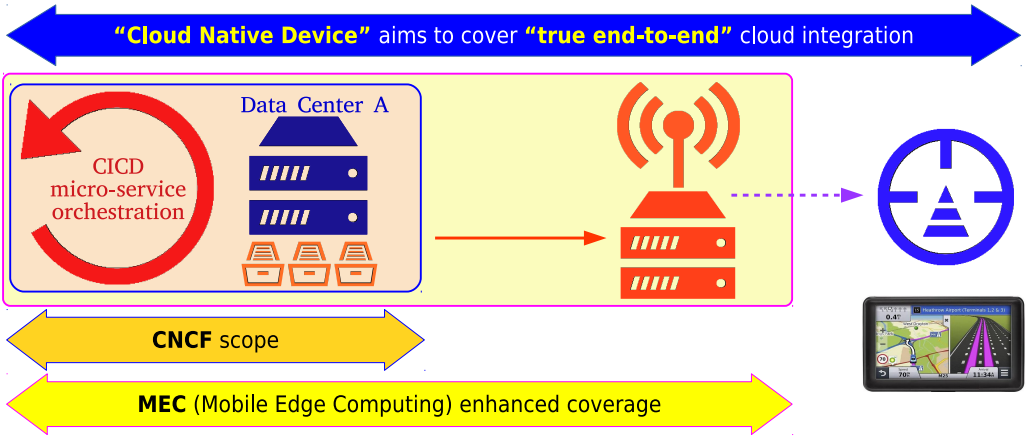
- センターサーバーに要求を出す
- サーバーのレスポンスを待つ
- この間、端末の操作はできない
- サーバーがユーザー要求を処理
- サーバーが結果を返す
- ユーザーは次の操作ができる

Cloud native = 非同期 (リアルタイム)

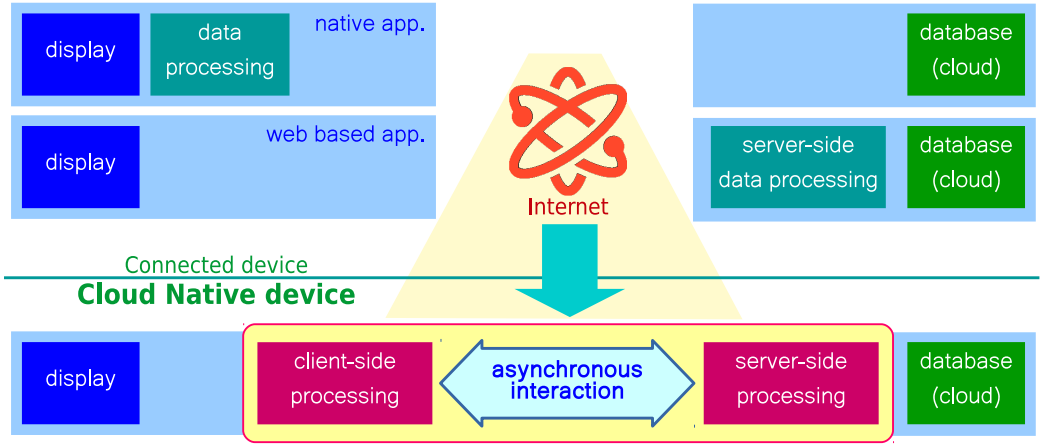
- センターサーバーに要求を出す
- 応答待ち期間中であっても,
  次のユーザー操作を受け付ける
- サーバーはユーザー要求を処理
- ユーザーは新たな処理を実行
- (可能であれば) クライアント内で
  代理応答を先に返す
- サーバーが処理結果を返す

「ストレスフリーな操作感」はクラウドとの非同期結合によってのみ実現可能な体験

これまで誰も語っていない "Cloud Native Device" コンセプト



次世代カーナビを実現させる (4) Cloud native 技術



"Could Native device" は最新の Web 技術で構成される

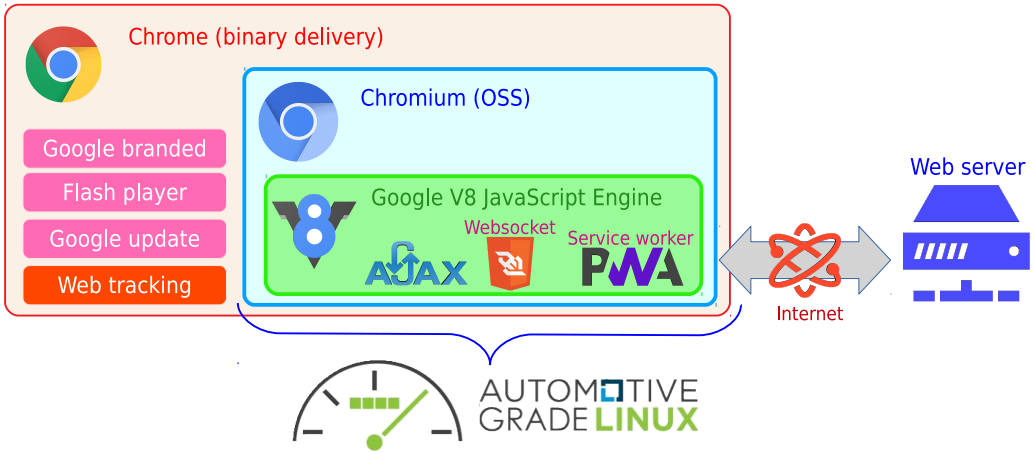
コネクテッドカーを実現可能な html5 (Web 技術) が充実してきた

Web アプリの根本概念 = コンテンツがそのまま色々な機器（環境）で実行できる

- Web アプリケーションを記述する方法
  - html5 = Web アプリケーションプラットフォーム（=コンテンツ記述言語）
  - Javascript = スクリプト言語
  - 標準 API = マルチメディア、グラフィックス、ファイル操作、通信、位置情報
    データベース、ストレージ、ウィジットなど多くの API が用意されている
  - 車載機向け API = VISS、VIAS など　← 車両の CAN 上に流れる情報の流通に対応
  - CSS = 表示修飾（=システム全体の Look & Feel を統一可能）
- Web アプリケーションの実行環境
  - Web Browser (PC, Mac, iOS, Android などの環境に予め用意されている)
  - Web Runtime (Browser を起動しなくても Web アプリケーションを実行できる)

車載機で html5 アプリケーションが実行できれば一気に世界が広がる可能性がある

Good news！: AGL には既に非同期通信技術が統合されている



三方にメリットあり,..　技術的に追及すべき方向なのは確かだろう

OEM の嬉しさ

- **クルマの魅力度アップ**
  (ユーザーの期待に応えられる)
- **クルマのSW管理が楽になる**
  * セキュリティ管理（更新）
  * プライバシー管理（制限）
  * サーバー側でのアプリ更新
- **Thin client (低コスト),**
  SW 更新で機能アップデート
- **新たな収益源になる可能性**
  サブスクリプション型の契約

ユーザーの嬉しさ

- **最新のデジタル体験ができる**
  クラウド環境の最適活用
- **カスタマイズ (パーソナライズ)**
  好みのアプリのインストール
- **常に最新の環境**
  (クルマは古くなっても最新SW)
- **使いやすい**
  スマフォなどで慣れた使い勝手

サービス事業者の嬉しさ

- **モビリティサービスプラットフォーム**
  クルマとクラウドの連携が可能に
- **プログラミング言語**
  JS, python など汎用言語の利用
- **アプリ連携（アグリゲーション）**
  マルチモーダル動作
- **ハードウエア非依存**
  アプリはどのHWでも動作可能

結　論

本日お伝えしたかったこと

- 既にバリバリ「スマフォ」を使い込んでいる現代のカーユーザー を満足させるには 単なる「コネクテッド」機能だけでは不十分 と考えています
- 解決手段の一案として、ドライバーとパッセンジャーに 真のデジタル体験の提供 を目指した "cloud native vehicle" のアイディア を紹介しました
- 真のデジタル体験の提供 を実現するには、既に約 10 年前に IT/Enterprise 領域で実用化されているクラウド／エッジ協調の技術（リアルタイム Web 技術） の導入検討から取り掛かる必要があると考えています
- このため「IT/Enterprise のクラウド技術」と「車載 ECU など組み込み機器の制御技術」の両方に明るいソフトウエア技術者の連携が不可欠、そのような企業や団体が"cloud native vehicle" を牽引していけるのだろうと考えています。